1. A method of simulating a logic design comprised of combinatorial logic and state logic, the method comprising:

representing the combinatorial logic and the state logic using separate graphic elements; and

associating computer code that simulates portions of the logic design with a graphic element that represents the combinatorial logic and with a graphic element that represents the state logic.

2. The method of claim 1, further comprising:

performing an error check on the graphic elements to determine if a single graphic element represents both combinatorial logic and state logic; and

issuing an error message if the single graphic element represents both combinatorial logic and state logic.

3. The method of claim 1, further comprising:

generating intermediate code that simulates the logic design; and

generating the computer code from the intermediate code.

4. The method of claim 1, wherein the computer code comprises one of C++ and Verilog.

5. The method of claim 4, wherein, if the computer code comprises C++, the method further comprises running the code through a cycle-based simulator to provide a simulation of the operation of the logic design.

6. The method of claim 4, wherein, if the computer code comprises Verilog, the method further comprises running the code through an event-driven simulator to provide a simulation of the operation of the logic design.

7. The method of claim 1, further comprising:

generating a topology of the logic design based on the graphic elements;

obtaining clock domains from the topology; and

generating the computer code based on the clock domains.

8. The method of claim 7, further comprising:

dividing the computer code into segments based on the logic cones; and

compiling the segments separately.

9. The method of claim 1, wherein state elements comprise elements which hold a particular logic state for a period of time and combinatorial logic elements comprise elements which combine two or more states to produce an output.

10. The method of claim 1, wherein the graphic elements comprise block diagrams.

11. An article comprising a machine-readable medium which stores executable instructions to simulate a logic design comprised of combinatorial logic and state logic, the instructions causing a machine to:

represent the combinatorial logic and the state logic using separate graphic elements; and

associate computer code that simulates portions of the logic design with a graphic element that represents the combinatorial logic and with a graphic element that represents the state logic.

12. The article of claim 11, further comprising instructions that cause the machine to:

perform an error check on the graphic elements to determine if a single graphic element represents both combinatorial logic and state logic; and

issue an error message if the single graphic element represents both combinatorial logic and state logic.

13. The article of claim 11, further comprising instructions that cause the machine to:

generate intermediate code that simulates the logic design; and

generate the computer code from the intermediate code.

14. The article of claim 11, wherein the computer code comprises one of C++ and Verilog.

15. The article of claim 14, wherein, if the computer code comprises C++, the article further comprises instructions that cause the machine to run the code through a cycle-based simulator to provide a simulation of the operation of the logic design.

16. The article of claim 14, wherein, if the computer code comprises Verilog, the article further comprises

instructions that cause the machine to run the code through an event-driven simulator to provide a simulation of the operation of the logic design.

17. The article of claim 11, further comprising instructions that cause the machine to:

generate a topology of the logic design based on the graphic elements;

obtain clock domains from the topology; and

generate the computer code based on the clock domains.

18. The article of claim 17, further comprising instructions that cause the machine to:

divide the computer code into segments based on the logic cones; and

compile the segments separately.

19. The article of claim 11, wherein state elements comprise elements which hold a particular logic state for a period of time and combinatorial logic elements comprise elements which combine two or more states to produce an output.

20. The article of claim 11, wherein the graphic elements comprise block diagrams.

21. An apparatus for simulating a logic design comprised of combinatorial logic and state logic, the apparatus comprising:

a memory that stores executable instructions; and

a processor that executes the instructions to:

represent the combinatorial logic and the state logic using separate graphic elements; and

associate computer code that simulates portions of the logic design with a graphic element that represents the combinatorial logic and with a graphic element that represents the state logic.

22. The apparatus of claim 21, wherein the processor executes the instructions to:

perform an error check on the graphic elements to determine if a single graphic element represents both combinatorial logic and state logic; and

issue an error message if the single graphic element represents both combinatorial logic and state logic.

23. The apparatus of claim 21, wherein the processor executes the instructions to:

generate intermediate code that simulates the logic design; and

generate the computer code from the intermediate code.

24. The apparatus of claim 21, wherein the computer code comprises one of C++ and Verilog.

25. The apparatus of claim 24, wherein, if the computer code comprises C++, the apparatus further comprises executes instructions that cause the machine to run the code through a cycle-based simulator to provide a simulation of the operation of the logic design.

26. The apparatus of claim 24, wherein, if the computer code comprises Verilog, the apparatus further executes instructions that cause the machine to run the code through an event-driven simulator to provide a simulation of the operation of the logic design.

27. The apparatus of claim 21, wherein the processor executes the instructions to:

generate a topology of the logic design based on the graphic elements;

obtain clock domains from the topology; and

generate the computer code based on the clock domains.

28. The apparatus of claim 27, wherein the processor executes the instructions to:

divide the computer code into segments based on the logic cones; and

compile the segments separately.

29. The apparatus of claim 21, wherein state elements comprise elements which hold a particular logic state for a period of time and combinatorial logic elements comprise elements which combine two or more states to produce an output.

30. The apparatus of claim 21, wherein the graphic elements comprise block diagrams.